Tuesday, 4 May 2021 9:02 PM

# String algorithms

size of text >> size of pattern

① Naive pattern matching

text → abracadabra
pat → cad

→ T[1 ... n]

n << N

Best case

→ t → aaaaaaaaaa
p → aaa

→ O(m)

Worst case

→ when all char of pat & text are same

t → aaaaaaaaaa
p → aaa

→ Instead of matching the whole pattern in the text, we compare the hash values

aabc → 13

text → aabcaadcadabra
pat → aabc

hash(aabc) = 13

38 ≠ 42

For each window

a
c
f
c

→ O(T)

→ How does this block box does its computations ??

Rabin Karp

Hash function

Hash value

abc → 294

If the hash function gives same hash value for two different strings, then it is the case of collision

* If for hash function, the frequency of collisions is very high, then it is not a good hash function.

Calculation of hash of a string

The most widely used way to define the hash value of a string is

of length m is,

hash(s) = s[0] · 256^(m-1) + s[1] · 256^(m-2) + ... + s[m-1] · 256^0 mod M

256 → 2

M → 101

m → chosen positive integer.

polynomial hash function

P or prime → is an integer larger than the input alphabet.

Eg P = 31

M → should be large as this increases the probability of more random strings to be indexing in 32 bit.

→ For all windows s = (T - P + 1) passing all windows in the block box.

→ more computations of block box = O(T)

→ Total TC → (T - P + 1) * P

→ (T * P)

Rabin Karp

→ you just pass the T window of text to the black box, and the new of text of all windows will be computed in O(1).

Rolling hash

HF

text

pattern

s = m

s = m-1

length (pattern) = m - 1

hash(txt[s+1 ... s+m]) = (d (hash(txt[s ... s+m-1]) - txt[s] * h) + txt[s+m]) mod q

Here, hash(txt[s ... s+m-1]): stack value at idx → s

hash(txt[s+1 ... s+m]): stack value at next index s+1

d → no. of character in the alphabet

h → pattern no.

h = d^(m-1)

→ length of pattern

text → "3 3 4 5 6"

pattern → 3 4

Since the length of pattern is 2, then the hash of text computation in hash function is also going to be 2

→ O(1)

37 * 10

370 + 5 = 375

---

① Calc the hash value of pattern

② Calc hash value of first window of text

→ O(T - P + 1) * 1

→ O(T + P - 1)

→ O(T)

375

HF → 5

375

Collisions

→ open addressing

→ chaining

→ Double hashing

→ (hash1(key) + i * hash2(key)) % size of hash table

hash2(key) = PRIME - (key % PRIME)

---

Given a m * n board,

Minimum SDC

words → ["oath", "pea", "eat", "rain"]

output → ["oath", "eat"]

words → ["abcd"]

output → ["abcd"]